EE379V: Assignment #3

Due: April 15 2022, 11:59pm TX

Introduction

Answer the following questions and explain solutions. Numbers in parentheses give maximum credit value. You can discuss in small groups, but turn in individual solutions and indicate collaborators. Do not use code from the Internet or high-level functions from other Python libraries (such as face detection or scene classification functions), unless specific permission is given.

Turn in assignments by Monday, April 15. Submit to Canvas (1) a .pdf with answers, descriptions, and figures; and (2) a .zip file containing your code without any large image or result files. The pdf can be created using latex or converting a word document to pdf or any other method you prefer, as long as it is organized and easy to read.

1 Recognition with Eigenfaces (PCA) (50%)

Figure 1: Problem 1

In this assignment you will implement the Eigenface and Fisherface methods for recognizing faces. You will be using face images from the Yale Face Database B where there are 10 faces under 64 lighting conditions (640 face images in total). Using your implementation, you will evaluate the ability of the algorithm to handle lighting conditions of the probe images that differ from those in the training images.

You can download the data for this assignment at: `http://www.cs.ucsd.edu/classes/sp05/cse152/faces.zip`

For more information on the Yale Face Database B, see: `http://vision.ucsd.edu/~leekc/ExtYaleDatabase/ExtYaleB.html`

Relevant papers:

- `https://sites.cs.ucsb.edu/~mturk/Papers/mturk-CVPR91.pdf`
- `https://cseweb.ucsd.edu/classes/wi14/cse152-a/fisherface-pami97.pdf`

From the set of all ten individuals, we will consider 5 subsets (of different lighting conditions), indexed as below:

Set 1. person*01.png to person*07.png (“brightest”)
Set 2. person*08.png to person*19.png
Set 3. person*20.png to person*31.png
Set 4. person*32.png to person*45.png
Set 5. person*46.png to person*64.png (“darkest”)
“*” indicates person’s id $(01, 02, \cdots, 10)$.

We have provided code ‘`readFaceImages.py`’ to read in the images and store the person labels and subset numbers for each image.

Write your code and report:

1) Take each 50×50 pixel training image and vectorize into a 2500-dimensional vector. Then perform principal component analysis (PCA) on the entire set of training image vectors, retaining the first d principal components. Remember to normalize your training vectors. The d eigenvectors (when reshaped and displayed as images) are the Eigenfaces. Display the top $d = 9$ eigenvectors (eigenfaces) after training. (15%)

2) For $d = 9$, on each subset, perform PCA and display one (your random pick) original and the corresponding reconstructed face, i.e., you will display $2 \times 5 = 10$ faces. (15%)

3) Train your Eigenface algorithm with $d = 9$ and $d = 30$ on all images in subset 1 (70 images). For each of your training images, project it to the d-dimensional Eigenspace. Classification will then be performed by nearest neighbor with the L2 (Euclidean distance) metric in the Eigenspace. Then, evaluate your algorithm on subsets 1-5, and report the error rates in a table (i.e., number of misclassified images over the total number of image, for each subset). For subset 1, you would expect perfect recognition because it is used for training. (20%)

Your final table should look something like this:

Table 1: Table to Report for Problem 1.

Subset	1	2	3	4	5
Error rate (%) $d = 9$					
Error rate (%) $d = 30$					

Useful functions include: `np.take_along_axis`, `np.linalg.svd` (or `np.linalg.eig`), `np.linalg.norm`, `plt.subplots`. You shall not use built-in PCA functions.

Acknowledgement: The data and basic design of this homework problem are from David Kriegman.

2 Scene Categorization (50%)

In the supplemental material, we have supplied **SIFT descriptors** from **8** outdoor scene categories: coast, mountain, forest, open country, street, inside city, tall buildings and highways. We only provide one example image, since you don't have to use the original images. Meanwhile, the full dataset can be downloaded here: `http://people.csail.mit.edu/torralba/code/spatialenvelope/`. SIFT descriptors of the dataset has been split into a train set (1888 images, `train_D.npy`) and test set (800 images, `test_D.npy`). The associated labels (integers from 1 to 8) are stored in `train_gs.npy` and `test_gs.npy`. Your task is to implement SIFT Bags-of-Words (BoW) features, train a nearest neighbor classifier on the training set, and evaluate the classifier on the test set:

- Implement K-means clustering algorithm to compute visual word dictionary. (10%)
- Build your visual Bags-of-Words as your image representation. (10%)
- Use nearest neighbor classifier to categorize them. (10%)

Describe and discuss your design decisions and report results:

- K-Means & BoW: Number of visual words, K-means stopping criterion, etc. (5%)
- Histogram & Classifier: binning parameters, K for KNN classifier, etc. (5%)
- Results: display the confusion matrix and report overall categorization accuracy (the percentage of test images that are correctly categorized). (10%)

Tips: If your feature computation is time consuming, you may consider using a low number of data points for the clustering. For example, you may use 10,000 SIFT descriptors to create a 100 word codebook.

Useful functions include: `np.histogram`, `scipy.spatial.distance.cdist`.

You *can* use built-in Kmeans and nearest neighbor classifier functions (e.g. `sklearn`).